Using Bittorrent and SVC for Efficient Video Sharing and Streaming

Amer Abdelhalim*, Toufik Ahmed†, Hidouci Walid-Khaled‡ and Satoshi Matsuoka*
* *Tokyo Institute of Technology, Tokyo, Japan*
amer@matsulab.is.titech.ac.jp, matsu@is.titech.ac.jp
† *CNRS LaBRI Lab. UMR 5800-University of Bordeaux-1, Talence, France*
tad@labri.fr
‡ *Ecole nationale Suprieure d'Informatique, Algiers, Algeria*
w_hidouci@esi.dz

***Abstract*—Massive and large scale content distribution over Internet is attracting a lot of research efforts as many challenges remain to be solved. Recent studies show that Internet video including video-to-TV and video calling is dominating the Internet traffic. As Internet becomes widely accessible to wired, mobile and wireless users, it is important to design a system that can ensure video streaming across variable network conditions while simultaneously handling devices and end-user heterogeneities.**

Most of the proposed solutions, such as CDN and peer-to-peer (P2P), solve the scalability problem but fail to handle receiver's heterogeneity. In this paper, we combine P2P network and SVC (Scalable Video Coding) to provide an efficient video sharing and streaming system. Our solution consists of an SVC layered extension of the widely used Bittorrent protocol to support real-time content delivery with different video qualities given the receivers capabilities. Thus, we propose different optimization techniques to organize peers in an overlay. The results, obtained by means of simulation, show that our system outperforms solutions that relay on single layer streams such as AVC (Advanced Video Coding) and this in terms of receivers perceived QoS.

***Keywords*-SVC; streaming; peer-to-peer; Bittorrent; QoS**

I. Introduction

Delivery multimedia services over the Internet, such as IPTV and video on demand (VoD), are becoming very popular and wide spread. According to [1], in the end of 2009, video content -excluding the one shared by P2P networks- has represented 33.2% of the consumed Internet traffic and would have approached 40% by the end of 2010 while a sevenfold increase in video traffic from 2009 to 2014 was predicted. Those services are QoS (quality of service) demanding and present many challenges due to use of best effort IP networks. Moreover, content consumers are accessing these services via disparate networks, such as broadband and wireless networks, and using a wide range of devices such as TV, computers, and smart-phones. Another aspect of this phenomenon is video sharing, where the user can contribute in generating and sharing video content. YouTube [2] falls into this category, where statistics show that YouTube represents 60% of the video watched on the Internet [3], and 20% of the HTTP traffic which is nearly 10% of all the traffic on the Internet [4].

Many solutions have been adopted in practice and proposed in the literature to tackle challenges related to real-time multimedia delivery. They vary from simple client/server streaming, to Content Delivery Networks (CDNs), peer-to-peer (P2P) systems, and cloud-based streaming. Network access, its dynamic variation, and terminal heterogeneity are posing more difficulties to guarantee an acceptable level of quality of service for the consumer. These issues have been addressed using different techniques ranging from simulcast of different qualities, to adapting the streams using Scalable Video Coding (SVC), Multiple Description Coding, Network Coding (NC), or their combination. Content adaptation can be even more efficient when carefully combined with cross-layer interactions. In this approach, network layers performances (link, network, and transport layers performance statistics), along with user preferences and terminal capabilities are gathered and used to coordinate the QoS adaptation at different levels of the protocol stack [5].

However, to provide a large-scale solution, P2P networks have emerged as an effective way of sharing large media content, leading to a more reliable and efficient system. They are used to overcome the deficiency of client/server solutions to provide large scale delivery and to manage the bandwidth bottleneck. Many P2P based video streaming systems have been developed and were successfully deployed. They are showing high QoS, scalability, and reliability compared to client/server models or CDN-based solutions. PPLive [6], Gridmedia [7] and Sopcast [8] are just few examples among others. Moreover, to ensure streaming over packet loss variable networks, such as mobile and wireless networks, characterized by a limited bandwidth and changing terminals capability, Scalable Video Coding (SVC) has been proposed to allow for a stream to be encoded with one or more substreams that can be decoded to offer different qualities [9]. Some work has been conducted in order to combine the features of SVC with the scalability, reliability and performance of P2P systems. In [10], a hybrid overlay network management for real-time multimedia streaming over P2P networks using SVC was proposed. Other authors tried to improve SVC with Multiple Description Coding

(MDC) [11] or Network Coding (ND) [12]. However, most of these works do not support the end-user capability to generate and share video content with different qualities. We believe that combining SVC and P2P systems can both help sharing and streaming video content with different qualities over variable network conditions and changing terminal capabilities. Both features are essential for large-scale and massive distribution of next generation services.

In this work, we propose an integrated solution combining the widely used Bittorrent protocol [13] and the multiple layers capabilities of SVC to provide different quality of service levels with a wide range of scalability. The cost and effort to deploy and prove such a large infrastructure in a real testbed can be time consuming and insurmountable. Thus, we propose to extend the packet-level ns-2 based Bittorrent simulator [14] to evaluate the effectiveness of our solution. The flexibility of using simulation let us innovate on peers and pieces selection algorithms, changing readily many important parameters and scaling to hundreds of peers while validating our results by using real video streams and network characteristics. Applying SVC streams in a P2P system increases the global utility of the content delivery, where we expect a better QoS compared to single-layer such as AVC stream. However, the straightforward extension exhibits a careful analysis as the peer organization into an overlay has an important impact on the delivered quality. To achieve a better QoS, we proposed to organize the overlay hierarchically by grouping together peers having the same capacity. In particular the peers having a high capacity will be placed near the source. The obtained results show that this strategy outperforms the Bittorrent peer organization. Furthermore, we studied the scalability issues of this solution in terms of offered QoS.

The rest of this paper is organized as follows: Section 2 presents the background necessary to understand this work including Bittorrent protocol and SVC standard. Section 3 describes our solution by giving the details on how we extended Bittorrent to stream scalable video considering real-time delivery constraints. Primary results are presented in Section 4 followed by a series of optimizations and their results. More results are presented in order to evaluate our system in terms of scalability. We discuss about related work in Section 5 and conclude in Section 6.

II. Background

A. Bittorrent protocol

Bittorrent [13] is a P2P file-sharing protocol designed to facilitate file transfers among multiple peers across unreliable networks. The protocol ensures the communication between the peers, seeders possessing the complete file as opposed to leechers, and a central entity, the tracker, which is used for peer discovery. The power of Bittorrent relies on basically two simple algorithms, choking and piece selection, proved to ensure high download rates for clients and avoiding free-riding behavior in the system. Choking controls to whom the peer can upload, and it is done for many reasons. It is designed in order to have good TCP performance, by limiting the number of unchocked peers, avoid fibrillation, by only changing choked peers once every ten seconds, and ensure good download rate for the peers, where each peer uses a tit-for-tat algorithm to reciprocate with peers who let it download. Additionally, the algorithm tries out unused connections every 30 seconds to find out if they might be better than the currently used ones, known as optimistic unchoking. The piece selection may be done randomly but can be improved by downloading first the rarest pieces. This can be approximated by knowing the pieces available at the neighborhoods.

Many researchers were interested in this powerful protocol and its possible application for video streaming purposes. In [15], the authors give higher download priority to pieces that are close to be used by the player (close to playback deadline), while a sliding window was used in [16]. Given the deployment difficulties of such a system, researchers preferred simulation over real environment. In [17] and [18], the authors implemented the original Bittorrent protocol and added some extensions using the OMNET++ simulator [19] to compare different streaming strategies. Another similar simulation was proposed on top of the ns-2 simulator [14], however implementing mainly the original Bittorrent specification and ignoring the details of the torrent files, like the hash values, and the tracker implementation. For simplicity reasons, we adopted the latter solution as a basis for our Bittorrent SVC streaming simulator.

B. Scalable Video Coding

Scalable Video Coding (SVC) [9] provides efficient video adaptation by truncating parts of the bitstream resulting in a valid substream. This feature is particularly useful when providing multimedia contents to different user terminal capabilities and network accesses, and variable network conditions. The H.264/SVC amendment is the last advances in this domain, which is a scalable extension of the H.264/AVC standard. Using SVC to encode video instead of AVC usually results in a slight increase in bitrate since SVC introduces encoding/decoding overhead. When encoding a bitstream using SVC, there has to be a trade-off between the available qualities, the desired bitrate, and the coding efficiency.

SVC bitstreams are composed of a base layer, compatible with H.264/AVC, which can be enhanced by one or multiple temporal, spatial or quality enhancement layers. The video is a set of GOPs (Group Of Pictures) containing frames called Access Units (AUs), which are composed of Network Abstraction Layer (NAL) units [9]. A NAL unit may belong to a certain layer which can be described as a triple (D,Q,T) representing its Dependency (spatial), Quality and Temporal level identifiers, where the higher levels depend on the

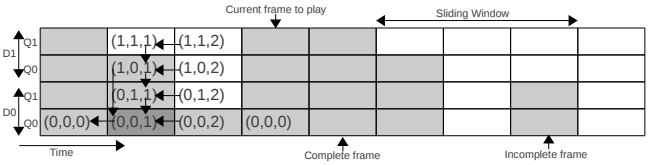

Figure 1. Snapshot of the buffer and the sliding window. The gray boxes are considered full (NAL units received) while the white ones are considered empty

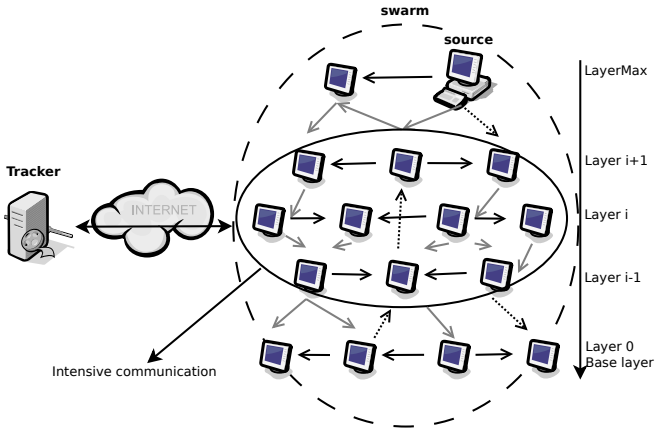

Figure 2. Overlay organization. The arrows represent the unchoked peers, where they go from senders to receivers. Unchoking priority order: 1) Black arrows when both involved peers are situated on the same layer. 2) Gray arrows between the next closest peers. 3) Dotted arrows for the rest.

lower ones in order to be decoded. Stream adaptation is achieved by dropping less important NAL units from the original bitstream. Examples of NAL unit dependencies are presented in the simplified buffer representation in Figure 1. The dependencies related to the dark gray NAL unit, (D,Q,T) = (0,0,1), are shown with arrows going from the higher level to the lower one. Obviously, if that NAL unit is not received, all the others that depend on it, directly or not, will not be able to be decoded. However, SVC is known to be resilient against packet loss, where the reconstructed video quality is proportional to the number of NAL units received, assuming priority is given to lower layers during the streaming process.

III. SVC BITTORRENT EXTENSION

In this work, we extend the previous Bittorrent protocol description to enable scalable video streaming. As in Bittorrent, there are two main entities in our solution: a tracker and peers, where a peer can be a seeder or a leecher. The overlay organization is depicted in Figure 2, and more details will be given in the next section. As a new peer joins the network, it contacts first the tracker in order to retrieve a list of peers having or consuming the same content, referred as a swarm. Then, it will establish a number of connections and start requesting pieces and distributing them after reception.

In P2P file-sharing systems, the client does not specify a constraint on the pieces or their reception order. However, this is the case in streaming systems. In addition, real-time constraints are imposed on the pieces which can be obsolete if not received at the scheduled interval in time. To respect those constraints, we use a sliding window to represent the next content to be played. This allows only pieces belonging to the sliding-window to be requested.

In Bittorrent, the source files are cut into pieces of the same size to enable concurrent downloading. However, in SVC, each NAL unit may belong to a specific layer, thus, peers can share the same NAL units, depending on the layers composing the desired video quality. Therefore, to exploit the potential of SVC, we will respect the video structure as described in Figure 1 along with the sliding window. This decomposition has the advantage of increasing the availability of the NAL units in our P2P system, since they are shared by the different substreams, and also the possibility to use an RTP packetization [20]. Such payload formatting will strengthen the delivery mechanism, especially under varying network conditions in terms of bandwidth and packet loss. It can also be used to adapt the stream between the end-points by using intermediate MANEs (Media Aware Network Elements) [9]. However this solution incurs an overhead due to the NAL unit size variation. To overcome this issue, a multi-NAL units messages [21] can be adopted in future work.

To take into account those considerations, we modified the piece selection algorithm as follows. The quality desired by the client is represented by a certain layer obtained by this formula: $Layer_i = D_i \times Q_{max} \times T_{max} + Q_i \times T_{max} + T_i$, where D_i, Q_i, T_i are the different spatial, quality, and temporal levels desired and Q_{max}, T_{max} are the maximum levels in the original video stream. In order to preserve the original Bittorrent algorithms, we included a test *Useful Piece* to the piece selection policies. A piece or an Access Unit is considered as useful, if it is not complete, belongs to the actual window, and the desired quality depends on it.

In a comparative study involving streaming extensions of Bittorrent [18], the authors described three different piece selection strategies: the *Fixed-Size Window* (FSW) that restricts the Bittorrent requested pieces within a sliding window; the *High-Priority Set* (HPS) uses a fixed-size set containing the next pieces in sequence to be downloaded, while pieces outside the set can be requested with a probability *p*; and the *Stretching Window* (SW) that combines the two previous approaches. The obtained results, in terms of QoS, showed that FSW and SW achieved comparable performance while outperforming the HPS strategy for a large window size. For the simplicity and the superior QoS achieved by the fixed-size window, we decided to use this policy with a large enough window size. Thus, the requested pieces are bounded by a window which is slided when completing the download of the lower bound frame, or receiving a time-out for this latter.

In the next section we present the evaluation and opti-

mization of this proposition.

IV. Evaluation and optimization

A. Test environment

As mentioned in the Section 2, we implemented our proposition on top of the ns-2 [22] Bittorrent simulator. To evaluate our solution we used the video *Elephant Dreams*, that can be found at [23], with *CIF* (Common Intermediate Format, 352x288 resolution) as a maximum resolution, having 15691 frames, a maximum framerate of *24* frames per second, and encoded using the JSVM Software[1]. We considered only the spatial and quality scalabilities in our tests, since the single layer AVC already supports temporal scalability. The SVC video was encoded with two spatial levels, referred to as CIF and QCIF (Quarter CIF, 176x144 resolution) and two different quality levels, Q0 and Q1, obtained using different quantization parameters as used in the examples of the JSVM Software. Each SVC layer is reflected in the AVC scenario by encoding a single-layer video with the same resolution and fidelity. The simulations were performed in a flash-crowd scenario where the peers arrive uniformly within a choking interval. We suppose a departure time of the peers based on an exponential distribution with parameter $\lambda = 1s^{-1}$. However, the network elements are generated following a star topology. A more realistic topology will be investigated in the future by using for instance the Georgia Tech Internet Topology Model (GT-ITM) [24].

Table I shows the distribution of peers bandwidths according to [25]. We assume a uniform distribution in [1,50]ms of the access links delay. These assumptions try to approximate the distributions observed for residential broadband networks [26]. We limited the simulation to broadband networks since, to the best of our knowledge, there is no extensive work on characterizing wireless and mobile networks which will be investigated in the future. To better evaluate our system, we assume a correlation between a peer's bandwidth and the corresponding desired video quality as described in Table II. Furthermore, for each quality we suppose there is only one seeder having a particular upload bandwidth according to Table II. We set the size of the window to 20 seconds, which represents 3% of the complete video, while an initial frame set of 2x the window size is prefetched to ensure a good start-up.

B. Initial evaluation

To evaluate our solution in terms of QoS, we measure the non-received NAL units ratio per leecher and report the resulting distribution when running the simulation with 40 peers. Measuring non-received NAL units is only indicative of the quality of the received substream because of the layer dependencies. However, in our solution we suppose that lower layers are always sent before enhancement layers and thus the number of NAL units received are correlated to the quality obtained by the end-user. A more precise measure, like the PSNR (Peak Signal-to-Noise Ratio) of the retrieved substream, will be considered in the future.

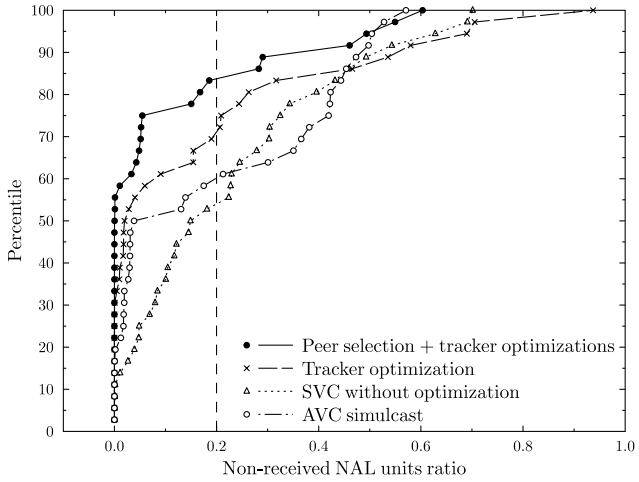

Figure 3. Cumulative distribution of non-received NAL units ratio when simulating a 40 peers network. The graph also shows the performance of AVC and the optimization results

In the SVC scenario, there is a single swarm where the peers with different qualities can exchange pieces, as opposed to the AVC scenario, where there are as many swarms as qualities without any communication between them. As a result, layered SVC increases the global utility of the content, and we expect a better QoS compared to single-layer AVC. We arbitrary define an acceptable QoS if the non-received NAL units ratio is less than or equal to 20% of the original stream. As we can see from Figure 3, in the case of SVC without optimization, half of the peers have a ratio greater than 20% and performs even worse than when using AVC, thus contradicting our expectations. From the point of view of overlay organization when using AVC, the peers are gathered into groups having the same quality resulting in a high reciprocation as opposed to when using SVC. Following these observations, we propose to organize the SVC overlay hierarchically by grouping together peers with the same capacity. Moreover, the hierarchical organization exploits the global availability of the SVC content by allowing the peers with different desired qualities to communicate. This overlay structure can be achieved at two levels: the tracker level, by returning the most suitable peer set to the requesting client; and on the peer level, by improving the peer selection algorithm.

C. Tracker optimization

The protocol between the peers and the tracker was not implemented. On the contrary, we used the tracker as a shared object between the peers. It is used at convenience to get the peer list and also to store global parameters and

[1]The version used is 9.19.14 and can be accessed with a command line CVS client:
cvs d :pserver:jvtuser:jvt.Amd.2@garcon.ient.rwth-aachen.de:/cvs/jvt login
cvs d :pserver:jvtuser@garcon.ient.rwth-aachen.de:/cvs/jvt checkout jsvm

Table I
PEERS CHARACTERISTICS AND THEIR DISTRIBUTION

Total upload bandwidth (Kbps)	256	320	384	448	512	640	768	1024	1500	3000
Contributed upload bandwidth (Kbps)	150	250	300	350	400	500	600	800	1000	1000
Download Bandwidth (Kbps)	512	640	768	1024	1300	2048	2048	3000	5000	9000
Distribution (%)	10.0	14.3	8.6	12.5	2.2	1.4	6.6	28.1	1.4	14.9

Table II
SEED AND QUALITY DISTRIBUTION

Download bandwidth range (Kbps)	<= 1024	] 1024 , 2048]	] 2048 , 5000]	> 4000
Seed upload bandwidth (Kbps)	500	1000	1500	2000
Desired quality	QCIFxQ0	QCIFxQ1	CIFxQ0	CIFxQ1

variables to limit the memory consumption of the simulation. As every peer consumes or provides a certain quality which in turn depends on a certain number of layers, the client's peer list will provide a large set of available layers. However, a random list of peers may not be able to provide a large set enough to satisfy all the layer dependencies. Thus, the peer list returned by the tracker should satisfy a maximum of these dependencies. Furthermore, SVC layers have a property where the closer the quality offered by two layers are the larger is the amount of shared content between them. As a result, gathering the peers with close desired layers together will result in a large content availability and a good reciprocation among them. The tracker can help to achieve this peer organization by returning a more adequate list of peers when receiving such request. More specifically, the tracker gives priority to the peers having the closest or equal layer to the requesting client's. However, despite two peers have close layers, there are cases where they may not be interested in each other. An example is when both peers have already shared all their content and are waiting for pieces that other peers with further layers may provide. To tackle this problem, we propose to let some peers (5 in our simulations) selected randomly regardless of their layer.

Figure 2 shows how we organize the overlay and how we manage the priority when unchoking. The peers requesting the same layer is considered as the optimal case where both peers share exactly the same NAL units. Hence, the more peers share content the more communication is high, as shown in the halo of intensive communication. Nevertheless, this solution is still not fair for the first peers that join the network. Since at the beginning there are few peers and we limit the number of connections (discussed further), they will receive the complete list and will be obliged to communicate with a non optimized set of peers. To address this issue, when a peer joins the network, it will first request a limited list (10 peers) and at the next interval it will request a normal sized one (25 peers). With this sacrifice of the first interval, all the peers will receive an optimized set at the next request. The positive results of the tracker peer list optimizations are shown in Figure 3. We can already observe that this overlay organization improves the QoS, and our SVC based solution outperforms the AVC case, but there is still room for improvement at the peer level.

D. Peer selection optimization

In Bittorrent protocol reciprocation algorithm, the receiver unchokes the peers with the highest download rate (upload rate is used in case of a seeder). If we consider a distant peer having a lower layer than the receiver's, as the peer shares more content, it gets a higher priority to be selected as an unchoking candidate. After the client receives all what the peer has to offer, a starvation situation can be reached if the other peers are selected in the same way because the higher layers can not be provided. Hence, a peer's layer is an important parameter that should be taken into account during the choking process. Therefore, as in the case of the tracker, layer priority is the first one to consider when ordering the peer list by the receiver, by unchoking the peers requesting the closest layer to the client's. The list is once again ordered according to the highest bandwidth and finally according to the service rate, to maintain a good reciprocation and avoid free-riding. Other parameters can be included to take into account, for example, the locality between the peers by using end-to-end round trip time (RTT) measurements [10]. However, the overlay is not totally organized due to the characteristics of the peer list returned by the tracker, and also because we only unchoke few peers, 4 in our case like in Bittorrent. Furthermore, peers can be unchoked without respecting these rules in the cases where there are only few peers or when using optimistic unchoking. The result of the final improvement is shown in Figure 3. If we take the same considerations concerning the limit for the non-received NAL units ratio for an acceptable QoS as in the initial evaluation, our solution can ensure the same quality for 83% of the peers against less than 60% when using AVC.

E. Scalability issues

In this section we analyze our system in terms of scalability, where scalability refers here to the growth of the system in terms of the number of peers. Figure 4 shows the non-received NAL units ratio when running the simulation with an increasing number of peers up to 100 peers. The graph shows clearly that the average QoS drops when increasing the number of peers. To better understand these results,

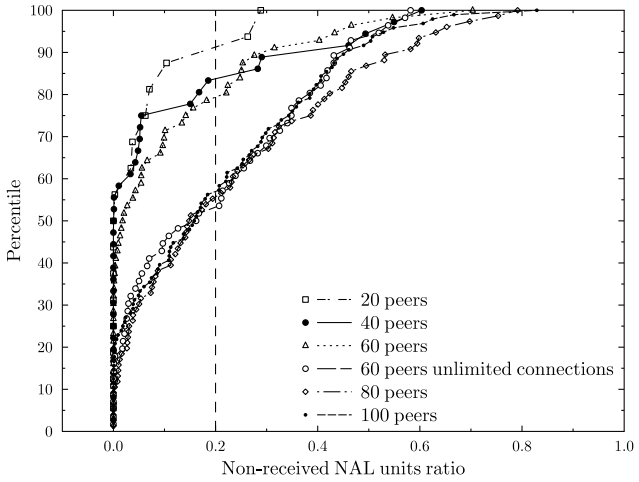

Figure 4. Cumulative distribution of non-received NAL units while increasing the number of peers

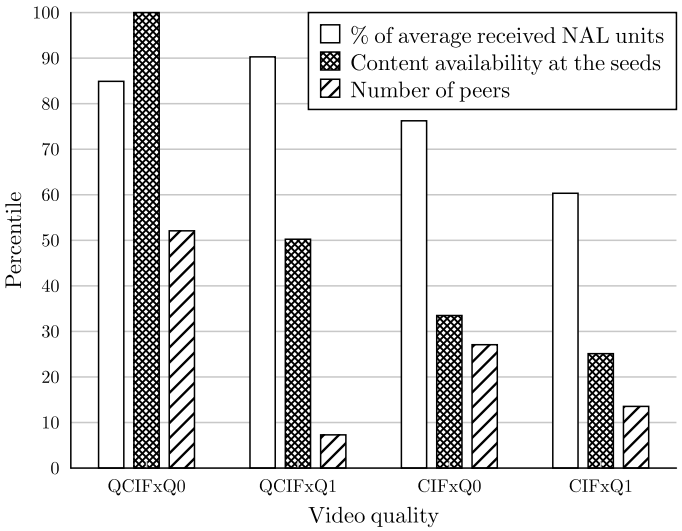

Figure 5. Percentage of average received NAL units, content availability at the seeds and the number of peers per quality for 100 peers. Note that, because every seed has the base layer, its content availability is 100%, thanks to SVC

we determined for 100 peers the percentage of average received NAL units for every quality along with their content availability at the seeders and the number of peers sharing them, and plotted the results in Figure 5. We can observe a correlation between the availability of the content at the sources and the respective downloaded NAL units ratio. However, for the peers with QCIFxQ1 quality, since they are few, the seeder and the higher quality peers can easily provide them the desired NAL units, thus getting the best QoS. We conclude that the observed drop of performance when scaling the number of peers is due mainly to the content availability, which is a well known limit of P2P systems.

Another important parameter to a achieve good scalability, in terms of QoS, is the number of connections per peer. The previous tests were obtained by limiting the number of connections to 30. In order to determine the effect of this parameter in our system, we don't limit the number of connections per peer when running the same simulation with 60 peers. The results are plotted in Figure 4. Obviously, having too many connections impair performance. This is because TCP congestion control behaves very poorly when sending over many connections at once and also because of the broadcasting of HAVE messages. Although the payload of this message is not large, encapsulated in TCP packets and broadcast every received NAL unit, it constitutes a large overhead [13]. Another optimization can be to avoid sending HAVE messages to a peer if it already have the piece [13] which will be investigated in the future.

V. Related work

Recently, many researchers were interested in combining SVC and Bittorrent like P2P systems for video streaming with high QoS. Broadcasting SVC videos over Bittorrent was proposed in [27]. In this work, the Bittorrent rarest-first policy was slightly modified where they also used a sliding window. However they only used quality scalability and did not fully exploit the SVC structure. Moreover, the overlay organization is the same as Bittorrent where layered SVC imposes a reorganization of the peers in order to improve the QoS. In [28], the authors successfully mixed SVC to a Bittorrent like P2P player and used a Prioritized Sliding Window to increase the QoS perceived by the user. Unlike our approach, their target was a real environment where they could run their tests across different networks and using different access devices. While this approach validates the use of the adaptability of SVC in real heterogeneous environments, it lacks scalability, where the system scaled at a maximum of 12 peers. Moreover, the peer selection does not exploit the SVC layers to organise the overlay and being vulnerable to free riding since no tit-for-tat policy was proposed. In [29] the authors proposed an unstructured self-adaptive P2P streaming solution, where they used a Bittorrent like mesh-pull mechanism. They also proposed a peer selection policy where each peer is laying on a layer and selects the peers situated on the same or higher layer, but does not exploit the content of lower layers and no tit-for-tat policy was used. Z. Liu et al. [25] proposed a Bittorrent like mesh-based open P2P streaming system that uses a tit-for-tat approach referred to as *Streaming Trading*, where *peers that upload more see higher quality video*. The system can accommodate different video coding schemes, like single-layer, layered, and multiple description coding, trading substreams rather than chunks, but without implementing them. Besides we followed the SVC structure instead of simple chunks, the *Stream Trading* approach can be interesting to compare to our piece selection algorithm.

VI. Conclusion and Future work

In this paper we proposed a Scalable Video Coding extension of the Bittorrent file-sharing protocol to provide

efficient video sharing and streaming over variable network conditions and end-user terminal capabilities. We adapted the protocol in order to support the real-time constraints imposed by the streaming process and the multi-layer SVC coding. Results showed a better QoS perceived by the end users compared to single layer AVC based solution. Packet level simulation proved to be flexible compared to real environment, while we could try many different approaches, algorithms and parameters and scaling easily to hundreds of peers without loosing underlying precision.

This work can be considered as a testbed for other solutions involving scalable video coding and Bittorrent base streaming systems. That is, we can extend this simulation to include for example Network Coding (NC) and Multiple Description Coding (MDC).

REFERENCES

[1] N. Leavitt, "Network-usage changes push internet traffic to the edge," *Computer*, vol. 43, no. 10, pp. 13 –15, oct. 2010.

[2] Youtube. [Online]. Available: www.youtube.com

[3] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, ser. IMC '07. ACM, 2007, pp. 15–28. [Online]. Available: http://dx.doi.org/10.1145/1298306.1298310

[4] Ellacoya data shows web traffic overtakes peer-to-peer (p2p) as largest percentage of bandwidth on the network. [Online]. Available: http://www.vaxination.ca/crtc/NXTcommEllacoyaMediaAlert.pdf

[5] I. Djama, T. Ahmed, A. Nafaa, and R. Boutaba, "Meet in the middle cross-layer adaptation for audiovisual content delivery," *Multimedia, IEEE Transactions on*, vol. 10, no. 1, pp. 105 –120, jan. 2008.

[6] Pplive. [Online]. Available: http://www.pplive.com/

[7] L. Zhao, J.-G. Luo, M. Zhang, W.-J. Fu, J. Luo, Y.-F. Zhang, and S.-Q. Yang, "Gridmedia: A practical peer-to-peer based live video streaming system," in *Multimedia Signal Processing, 2005 IEEE 7th Workshop on*, 30 2005-nov. 2 2005, pp. 1 –4.

[8] Sopcast. [Online]. Available: http://www.sopcast.com/

[9] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h.264/avc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007. [Online]. Available: http://dx.doi.org/10.1109/TCSVT.2007.905532

[10] M. Mushtaq and T. Ahmed, "Hybrid overlay networks management for real-time multimedia streaming over p2p networks," in *Real-Time Mobile Multimedia Services*, ser. Lecture Notes in Computer Science, D. Krishnaswamy, T. Pfeifer, and D. Raz, Eds. Springer Berlin / Heidelberg, 2007, vol. 4787, pp. 1–13.

[11] F. de Asís López-Fuentes, "P2p video streaming combining svc and mdc," *Applied Mathematics and Computer Science*, vol. 21, no. 2, pp. 295–306, 2011.

[12] S. Mirshokraie and M. Hefeeda, "Live peer-to-peer streaming with scalable video coding and networking coding," in *Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, ser. MMSys '10. New York, NY, USA: ACM, 2010, pp. 123–132. [Online]. Available: http://doi.acm.org/10.1145/1730836.1730852

[13] Bittorrent protocol specification. [Online]. Available: http://wiki.theory.org/BitTorrentSpecification

[14] K. Eger, T. Hoßfeld, A. Binzenhöfer, and G. Kunzmann, "Efficient simulation of large-scale p2p networks: packet-level vs. flow-level simulations," in *Proceedings of the second workshop on Use of P2P, GRID and agents for the development of content networks*, ser. UPGRADE '07. New York, NY, USA: ACM, 2007, pp. 9–16. [Online]. Available: http://doi.acm.org/10.1145/1272980.1272986

[15] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "Bitos: Enhancing bittorrent for supporting streaming applications," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, april 2006, pp. 1 –6.

[16] P. Shah and J.-F. Paris, "Peer-to-peer multimedia streaming using bittorrent," in *Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa*, april 2007, pp. 340 –347.

[17] K. Katsaros, V. Kemerlis, C. Stais, and G. Xylomenos, "A bittorrent module for the omnet++ simulator," in *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*, sept. 2009, pp. 1 –10.

[18] C. Stais, G. Xylomenos, and A. Archodovassilis, "A comparison of streaming extensions to bittorrent," in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, 28 2011-july 1 2011, pp. 1068 –1073.

[19] Omnet++. [Online]. Available: www.omnetpp.org

[20] Rtp payload format for scalable video coding. [Online]. Available: http://tools.ietf.org/html/draft-ietf-avt-rtp-svc-27

[21] S. Medjiah, T. Ahmed, E. Mykoniati, and D. Griffin, "Scalable video streaming over p2p networks: A matter of harmony?" in *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2011 IEEE 16th International Workshop on*, june 2011, pp. 127 –132.

[22] ns-2 network simulator. [Online]. Available: http://isi.edu/nsnam/ns/

[23] Yuv sequences. [Online]. Available: http://trace.eas.asu.edu/yuv/

[24] Georgia tech internet topology model (gt-itm). [Online]. Available: http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html

[25] Z. Liu, Y. Shen, K. Ross, S. Panwar, and Y. Wang, "Substream trading: Towards an open p2p live streaming system," in *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, oct. 2008, pp. 94 –103.

[26] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu, "Characterizing residential broadband networks," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, ser. IMC '07. New York, NY, USA: ACM, 2007, pp. 43–56. [Online]. Available: http://doi.acm.org/10.1145/1298306.1298313

[27] V. R. P.Garca Ortiz, J.M. Dana and I.Garca, "Broadcasting of h.264/svc video over bittorrent-like networks," *Actas del Workshop on MultimediaData Coding and Transmission (WMDCT)*, p. pages 41 46, 2010.

[28] M. S. N. Roberto Pontes Nunes, Rui Santos Cruz, "Scalable video distribution in peer-to-peer architecture," in *CRC'2010 - 10a Conferencia sobre Redes de Computadores*, 2010, pp. 95–100.

[29] P. Baccichet, T. Schierl, T. Wiegand, and B. Girod, "Low-delay peer-to-peer streaming using scalable video coding," in *Packet Video 2007*, nov. 2007, pp. 173 –181.